Demo paper: Tablet-based visualization of transportation data in Madrid using SPARQLStream

Jean-Paul Calbimonte, Alejandro Fernández-Carrera, Oscar Corcho

Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
jp.calbimonte@upm.es alej4fc@gmail.com ocorcho@fi.upm.es

Abstract. In this demo paper we describe an iOS-based application that allows visualizing live bus transport data in Madrid from static and streaming RDF endpoints, reusing the Web services provided by the bus transport authority in the city and wrapping them using SPARQL$_{\text{Stream}}$.

1 Introduction

Data from public administration and services in countries, regions and cities are increasingly made publicly available. However, most of this data is produced and exposed in very heterogeneous formats, represented in different models and deployed under diverse technologies. While developers can take advantage of this data, the time taken in understanding the models and formats constitutes a barrier to enable the integration of the different data sources needed for a given application. The use of standard vocabularies and data models such as RDF and query languages as SPARQL have proven to be helpful to overcome this semantic heterogeneity.

This is also the case with public transportation data in a large city such as Madrid. The local bus transport authority, EMT, provides a set of web services[1] that allow retrieving information such as the bus stops, bus lines, routes, stop locations, and other mainly static data. The data is returned by these services as XML documents. For example the following XML response in Listing 1.1 provides the details about a bus stop.

```
<Stop>
  <IdStop>28</IdStop><PMV>61247</PMV>
  <Name>P CASTELLANA-JUZGADOS</Name>
  <PostalAdress>P de la Castellana, 187 (Juzgados)</PostalAdress>
  <CoordinateX>-3.68972639781606</CoordinateX>
  <CoordinateY>40.46506045830l5</CoordinateY>
</Stop>
```

Listing 1.1. EMT XML response: Bus stop details.

This type of information is mainly static and does not change too often. Therefore we can use standard RDF converters to transform them into triples and store them in a triple store, using an ontological model. Nevertheless, more rapidly-changing data is also available in the EMT services, such as the current waiting times in every bus stop in the city. Current RDB2RDF[2] tools do not allow querying such dynamic data

[1] EMT Services: https://servicios.emtmadrid.es:8443/geo/servicegeo.asmx
[2] RDB2RDF Working Group: http://www.w3.org/2001/sw/rdb2rdf/

that can be seen as data streams. Moreover, most RDF stream processing engines lack the flexibility of defining mappings between the source schemas and the ontological schema. Of these, only SPARQL$_{\text{Stream}}$ supports using declarative R2RML mappings[3] for doing so. Furthermore, current implementations of RDF stream query engines are hard to access for end-user applications.

In this demo we show how we enabled an iOS tablet application to access both the static and dynamic data from the city of Madrid, and in particular the data from the EMT web services. We showcase the use of an RDF stream SPARQL endpoint (using the Morph-streams[4] implementation of SPARQL$_{\text{Stream}}$), that can be accessed by the client application in the same way it accesses a static SPARQL endpoint.

2 Architecture & Implementation

The architecture of the system is the following (Figure 1): the client application is a Tablet iOS App that requests static data from a SPARQL endpoint that contains Bus station data including name, geo-location, etc. With this data, the App can display the bus stops in a Google Map. The App can also send queries and receive results from a live SPARQL$_{\text{Stream}}$ endpoint that provides data about the reported waiting times in each bus stop. The endpoint is implemented using Morph-streams, that serves virtual RDF streams through SPARQL$_{\text{Stream}}$ continuous queries.

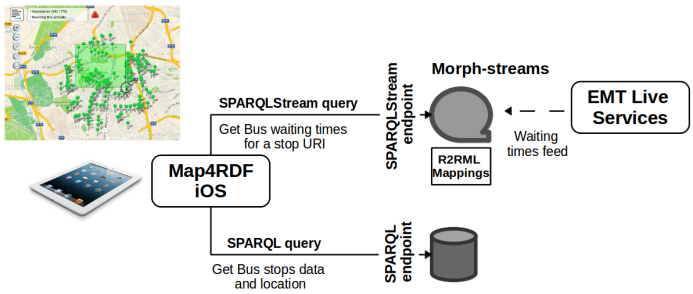

Fig. 1. System Architecture: The App and the SPARQL and SPARQL$_{\text{Stream}}$ enpoints.

2.1 SPARQL$_{\text{Stream}}$ queries

The SPARQL$_{\text{Stream}}$ endpoint runs a live virtual RDF stream that is directly fed from the Madrid EMT web services. SPARQL$_{\text{Stream}}$ was first introduced in [1], and has been inspired by previous proposals of streaming-oriented extensions of SPARQL, mainly C-SPARQL[2]. SPARQL$_{\text{Stream}}$ is based on the concept of virtual RDF streams of triples that can be continuously queried, and whose elements can be bounded using sliding windows. The SPARQL$_{\text{Stream}}$ syntax follows closely that of SPARQL 1.1, adding window constructs for RDF stream graphs and additional solution modifiers.

[3] R2RML W3C Recommendation: `http://www.w3.org/TR/r2rml/`
[4] Morph-streams: `https://github.com/jpcik/morph-streams`

In SPARQL$_{\text{Stream}}$, each virtual RDF stream graph is identified by an IRI, so that it can be used or referenced elsewhere in the query, and time windows of the form `[start TO end SLIDE slide]` can be applied to it. As an example, the query in Listing 1.2 requests the bus waiting times per stop reported in the last 5 minutes, from the `http://emt.linkeddata.es/data#busstops.srdf` stream graph.

```
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt: <http://data.nasa.gov/qudt/owl/qudt#>
PREFIX emt: <http://emt.linkeddata.es/data#>
SELECT ?waittime ?obs ?stop
FROM NAMED STREAM <http://emt.linkeddata.es/data#busstops.srdf> [NOW - 300 S]
WHERE {
  ?obs a emt:BusObservation;
      ssn:observedBy ?stop.
      ssn:observationResult ?result.
  ?result emt:timeToBusValue ?av.
  ?av qudt:numericValue ?waittime.
}
```

Listing 1.2. SPARQL$_{\text{Stream}}$ query requesting waiting times reported in the latest 5 min.

2.2 R2RML Mappings

As we stated before, RDF streams in the system are virtual, so we use mappings to relate the ontological terms of the SPARQL$_{\text{Stream}}$ queries to the schema of the original EMT stream. for this, Morph-streams uses an R2RML set of mappings, as described in [3]. For example, the following mappings show how to relate the `TimetoBusValue` ontological concept with the EMT stream (including how to construct the URI).

```
:timeToBusValue a rr:TriplesMap; rr:logicalTable :emtStream;
  rr:subjectMap [ rr:template "http://transporte.linkeddata.es/emt/busstop/id/{stopid}/
    busline/{lineid}/timeToBusValue/{timed}";
  rr:class emt:TimeToBusValue; rr:graph emt:busstops.srdf ];
  rr:predicateObjectMap [rr:predicate qudt:numericValue;
                         rr:objectMap [rr:column "timetobus"]];.
```

Listing 1.3. R2RML sample mappings.

2.3 The Tablet application

The Map4RDF iOS App is designed as a general purpose RDF-based application for mobile devices, implemented in Objective-C and targeted to display geo-referenced information from SPARQL endpoints. In this way, the developer needs only to provide the suitable SPARQL queries that fulfill his data needs, and get the results in JSON format. In a first stage the App identifies data categories in a static SPARQL endpoint that can be displayed in a map (e.g. instances of `emt:BusStop`). The App is able to recognize latitude-longitude coordinates using the WGS84 Geo Positioning vocabulary[5] and also WKT points and polygons. The App already has implemented usual features such as layering, bounding box views, clustering of visualized points, etc. that can be applied to the visualized points.

Once the geo-referenceable instances are identified (through their URIs), the App can be configured to retrieve more information from each of these instances when the user clicks over it. In the case of the Bus stop application, it can retrieve the stop name

[5] `http://www.w3.org/2003/01/geo/wgs84_pos#`

and description from a static SPARQL endpoint, and the bus waiting times from a streaming SPARQL$_{\text{Stream}}$ endpoint. For instance, the App can easily display a bus route in the map as in Figure 2. The App also natively recognizes statistical data if available in the RDF dataset (supports the RDF Data Cube Vocabulary[6]). One of the key features of Map4RDF is that the data retrieval is entirely configurable by the developer, by specifying the SPARQL and SPARQL$_{\text{Stream}}$ queries and endpoints.

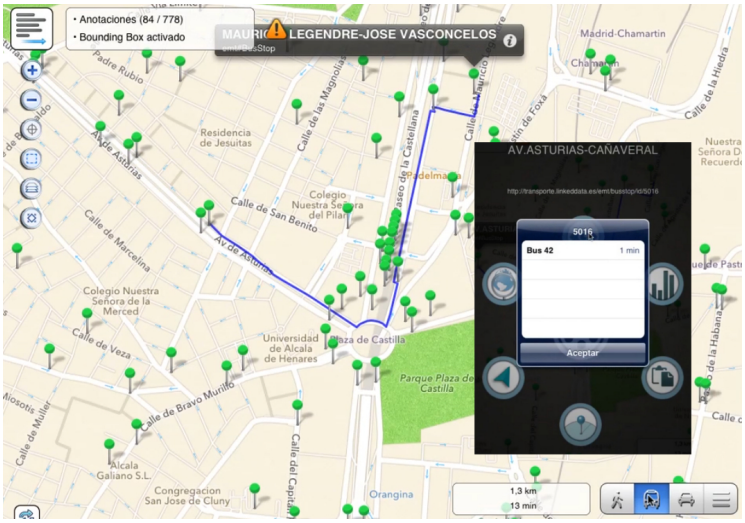

Fig. 2. Bus route displayed in a map from an origin stop to a destination, and also displaying the waiting time in a particular bus stop.

The App is able to relate the static data and the dynamic data thanks to the URIs that link both datasets: e.g. A Bus stop URI in the static endpoint, is referenced by a bus waiting time observation in the streaming endpoint. In this way if the user clicks on a bus stop, a query is launched against the SPARQL$_{\text{Stream}}$ endpoint, requesting the current waiting times for the stop represented by that URI (Figure 2).

Acknowledgements

This work has been supported by the PlanetData FP7 257641 Project. We thank EMT Madrid for providing access to their data services.

References

1. Calbimonte, J.P., Corcho, O., Gray, A.J.G.: Enabling ontology-based access to streaming data sources. In: Proc. 9th International Semantic Web Conference. (2010) 96–111
2. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: C-SPARQL: A continuous query language for RDF data streams. International Journal of Semantic Computing **4**(1) (2010) 3–25
3. Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. Int. J. on Semantic Web and Information Systems **8** (2012) 43–63

[6] `http://www.w3.org/TR/vocab-data-cube/`